NUCLEONICA SCRIPTING

R. DREHER, J. MAGILL, A. BERLIZOV
European Commission, Joint Research Centre,
Institute for Transuranium Elements, Postfach 2340, 76125 Karlsruhe, Germany

C. GRAMMES, C. ZANG
DIaLOGIKa GmbH, Pascalschacht 1, D-66125 Saarbrücken – Germany
Dokumenta S.A.16, rue d'Epernay, L-1490 Luxembourg - Luxembourg

ABSTRACT

An overview of the Nucleonica scripting language and module is presented. A detailed case study on targeted alpha therapy is described which demonstrates powerful features of Nucleonica for modelling experiments through the use of batch scripts.

1. Introduction

Nucleonica applications are designed to be very user friendly, intuitive, and require a minimum of learning time [1,2]. For users, who prefer a more "hands on" approach, Nucleonica provides this with its scripting module. Scripting refers to a programming task, in which pre-existing components or applications are connected to accomplish a new task. In accordance with this definition, the Nucleonica scripting gives the user a powerful programming interface through which he/she can access basic nuclear data and run all the Nucleonica applications.

2. The Scripting Language in Nucleonica

The Nucleonica scripting is an easy-to-use physically intuitive object oriented programming language. The basics of the language, such as variable types (*int*, *double*, *string* or arrays), logical (&&, ||) and mathematical (+, -, *, /, %) operators, control structures (*for* and *while* loops, *if* statements) and built-in functions (*sin*, *log*, *print*, *abs*, *exp*, *sqrt, pow* etc.), are described in detail in the respective section of the Nucleonica wiki [3]. This wiki also serves as an on-line reference on the features of the implemented classes, their properties and methods. With ready-to-use examples and pre-defined scripts, which are largely enriched with comments and explanations, users can easily learn the language and save the time when developing their own scripts. The basic components of the language (classes) are summarised in Table 1.

Class	Main methods	Related applications	Related classes
nuclide, vnuclide	*Decay()*	*Decay Engine*	*decayResult*
	Dosimetry()	*Dosimetry and Shielding*	*doseResult*
	DecayInfo()	*Nucleonica database*	*decayInfo*
	Radiations()	*Nucleonica database*	*radiations*
range	*CalculateMono()*	*Range and Stopping Power*	*rangeResult*
	CalculateCompound()	*Range and Stopping Power*	*rangeResult*
transport	*Calculate()*	*Transport and Packaging*	*transportResult*
korigen	*callKorigen*	*webKORIGEN*	*korigenResult*

Tab 1: Basic components of the Nucleonica scripting language

In the script, the classes are instantiated through the objects which provide access to specific properties (*object.property*) and methods (*object.method(parameterlist)*). The *nuclide* class is of the central importance in the Nucleonica scripting, as it gives access to all basic properties of nuclides and implements the most important Nucleonica applications, such as “Decay Engine” and “Dosimetry and Shielding”. The *nuclide* class is instantiated in two steps. First, the *nuclide* object is declared and, then, its properties are initialized using the *Create()* method:

```
nuclide Th232;                  // Declare the nuclide object with name Th232
Th232.Create("Th", 232, 0);     // Get properties of Th-232 from the Nucleonica database
```

Three parameters of the *Create()* method define a nuclide of interest, i.e. specific element, isotope and isomeric state (0 refers to the nuclide's ground state). Once the *nuclide* object is created, one can use its properties and methods to access and manipulate the related nuclear data. For example, the *Th232.Halflife* property returns the half-life of ^{232}Th, and the *Th232.Activity(amount, fromQuantity, toQuantity)* method converts given amount of ^{232}Th between different quantities (activity, mass, number of atoms) and measurement units (e.g. Bequerels, Curies).
Using the *print()* function, which accepts a string as an input parameter, one can readily display the nuclide properties in the script output window. As shown in the following examples, the string to be displayed can be composed by the concatenation of ordinary strings and implicitly or explicitly (using *format()* function) converted numerical values (“+” serves as the concatenation operator):

```
// the integer atomic number Z is converted implicitly to a character string
print("Atomic number of Th-232 = " + Th232.Z);
// the explicitly formatted atomic mass (AWR_C12) shows 3 digits after the decimal point
print("Atomic mass of Th-232 = " + format(Th232.AWR_C12, "0.000") + " a.m.u.");
// the decay constant (natural log(2) divided by the half-life) is shown in scientific notation
print("Decay constant of Th-232 = "+format(log(2)/Th232.Halflife,"0.00E+00")+" per sec");
```

More specific nuclear data can be retrieved from the Nucleonica database using the *nuclide.DecayInfo(daughterIndex)* and *nuclide.Radiations(RadiationType)* methods. The *DecayInfo()* method retrieves decay data for a specified daughter nuclide in the nuclide’s decay chain, which are returned as properties (such as *BranchingRatio*, *Energy, DecayMode*) of the *decayInfo* object. The *Radiations()* method returns the list of energies and per-decay emission probabilities for the decay radiations of the requested type (α, β, γ, n, p, spontaneous fission fragments, discrete energy electrons, X-rays or annihilation photons) emitted in the decay of the given nuclide.
Among other *nuclide* object methods, the *nuclide.Decay()* method allows the user to perform radioactive decay calculations. Through its extensive lists of input and output parameters, the method provides the full functionality and flexibility of the respective Nucleonica Decay Engine application. The input parameters include the amount of the nuclide, the length of the time interval, the number of time steps, the accuracy of the calculation, and 15 additional options specifying the list of the target quantities (such as, gamma emission rate, ingestion and inhalation radiotoxicities, α, $\alpha+\beta$, and $\alpha+\beta+\gamma$ isotopic powers etc.), which are required in the calculation output. The results of the decay calculations are returned in the *decayResult* object, containing an array of the cumulative (e.g. the total activity and radiotoxicity of the parent nuclide and all its decay products) and nuclide specific (for the parent and each daughter nuclide) data, evaluated for the specified points of time.
Whereas the *nuclide* represents a real physical object, the *range* object corresponds to setting up an experiment. Through the *range.CalculateMono()* method, one can calculate the range and stopping power for the specified projectile type and single element target. To define the "experimental setup", the method uses 7 parameters, namely: 1) the type of the projectile (e.g., 2 – protons, 7 – other ions); 2) the projectile material index, if projectile type is 7; 3) the projectile energy; 4) the energy units (e.g., 0 means MeV); 5) the atomic number of the target element; 6) the density of the target material; and 7) the physical state of the target

(e.g., 0 - for solids and liquids). The results of the calculation are returned in the *rangeResult* object, which, first, has to be created, as shown in the following example:

```
range ProtonsInTh;                    // Declare the range object for protons in thorium
rangeResult Range_Results;            // Declare the rangeResult object
// Calculate the range and stopping power for 33.5 MeV protons in the thorium target
Range_Results = ProtonsInTh.CalculateMono(2,0,33.5,0,Th232.Z,Th232.Density,0);
```

In addition to the built-in functions, user-defined functions, e.g. for numerical integration, can be included in the script. The scripts can be directly entered and run in the Nucleonica source editor page. Scripts can be also developed locally on the user's computer and then uploaded, stored and run in the server. The stored scripts can be further modified, downloaded or distributed to other users.

3. Case Study: A Nucleonica script for ^{230}U production optimization

The isotope pair ^{230}U/^{226}Th has been recently identified [4] as being suitable for targeted alpha therapy. In the current case study, a Nucleonica script is described, which models the production of ^{230}U through the proton irradiation of natural thorium, as shown in Fig.1.

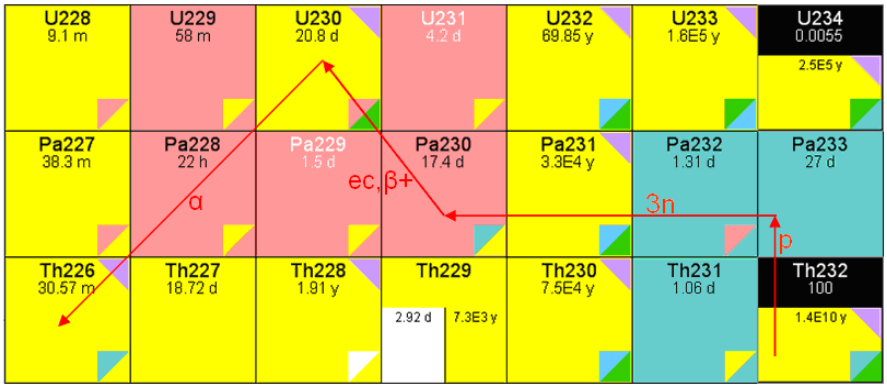

Fig.1. ^{230}U/^{226}Th production scheme: ^{230}U is accumulated in ε/β^+-decay of ^{230}Pa, produced in (p,3n)-reaction on ^{232}Th (graphic from the Nucleonica Nuclide Explorer).

In the following sections the production steps are described in detail and illustrated by the most significant lines of the script. The full script is available on the Nucleonica wiki [3].

3.1 Modelling the production of ^{230}Pa

At this stage of the experiment, the thickness of a thorium foil is the most important parameter. It determines the amount of ^{230}Pa produced at the end of the irradiation and influences the efficiency of the subsequent radiochemical separation. From the threshold behaviour of the ^{232}Th(p,3n)-reaction (see Fig.2), one can infer the lower limit of the useful energy range of protons to be $E_{min} \approx 15$ MeV. Assuming $E_{max} = 33.5$ MeV for the energy of the incident protons, the estimate for the foil thickness is then $X_{foil} = R_p(E_{max}) - R_p(E_{min}) \approx 2.23$ mm – 0.58 mm = 1.65 mm, where $R_p(E)$ is the range of protons with energy E in thorium. The respective Nucleonica script looks as follows:

```
Range_Results = ProtonsInTh.CalculateMono(2,0,33.5,0,Th232.Z,Th232.Density,0);
double Emax_Range; Emax_Range = Range_Results.ProjRange;
Range_Results = ProtonsInTh.CalculateMono(2,0,15.0,0,Th232.Z,Th232.Density,0);
double Emin_Range; Emin_Range = Range_Results.ProjRange;
double FoilThickness; FoilThickness = Emax_Range - Emin_Range;
```

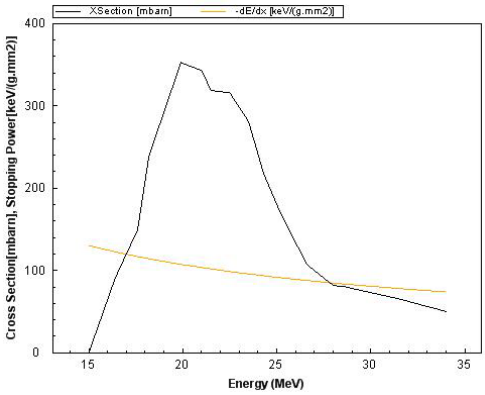

Fig.2. The experimental cross-section $\sigma(E)$ for the reaction ^{232}Th(p,3n)^{230}Pa [4] and the calculated stopping power $S(E)$ for protons in thorium as functions of the proton energy.

To calculate the amount of ^{230}Pa produced by the end of the irradiation, one needs first to evaluate the average cross-section $\langle\sigma\rangle$ for the (p,3n)-reaction. Assuming constant proton flux inside the target, one can obtain the following formula:

$$\langle\sigma\rangle = \frac{1}{X_{foil}}\int_0^{X_{foil}}\sigma(x)dx = \frac{1}{\rho X_{foil}}\int_{E_{max}}^{E_{min}}\sigma(E)\frac{d(\rho x)}{dE}dE = \frac{1}{\rho X_{foil}}\int_{E_{min}}^{E_{max}}\frac{\sigma(E)}{S(E)}dE = 163\ mb$$

where ρ - the target density, $\sigma(E)$ - the reaction cross-section and $S(E) = -dE/d(\rho x)$ - the proton stopping power as functions of proton energy E (see Fig.2). Below is a part of the user-defined function *AverageXSection*(), which implements this formula using a trapezoidal integration:

```
double AXS = 0;                                   // Average cross-section
for (i = 1; i < 16; i += 1)                       // Loop on the proton energies
{  Range_Results = ProtonsInTh.CalculateMono(2,0,pE[i-1],0,Th232.Z,Th232.Density,0);
   double F_Emin; F_Emin = XSection[i-1] / Range_Results.STotal;
   Range_Results = ProtonsInTh.CalculateMono(2,0,pE[i],0,Th232.Z,Th232.Density,0);
   double F_Emax; F_Emax = XSection[i] / Range_Results.STotal;
   AXS += (F_Emin + F_Emax) * (E[i] - E[i-1]) / 2;  }
AXS = AXS / (FoilThickness*Th232.Density);     // Final normalization
```

Here, *E*[] and *XSection*[] are the proton energy and reaction cross-section arrays respectively, *STotal* is the property of the *rangeResult* object giving the value of the total stopping power. Having evaluated the average cross-section, one can easily calculate the production rate of ^{230}Pa as $R_{Pa\text{-}230} = N_{Th}\langle\sigma\rangle X_{foil} I = 1.0\cdot10^{12}$ s^{-1}, assuming the atomic density $N_{Th} = 3.04\cdot10^{22}$ cm^{-3} and the proton current I = 200 μA.
Once the production rate of ^{230}Pa is known, the amounts of ^{230}Pa and ^{230}U, obtained by the end of the irradiation period, can be evaluated. Using Bateman's analytical solution for the system of differential equations, governing the nuclide decay and buildup, one can obtain:

$$A_{Pa}(t) = R_{Pa-230}(1-e^{-k_{Pa}t}), \qquad A_U(t) = R_{Pa-230}\frac{k_U\cdot k_{Pa}\cdot BR_{Pa,U}}{k_U - k_{Pa}}\left(\frac{1-e^{-k_{Pa}t}}{k_{Pa}} - \frac{1-e^{-k_U t}}{k_U}\right),$$

where $A_{Pa}(t)$ and $A_U(t)$ - the activities of ^{230}Pa and ^{230}U at time t, k_{Pa} and k_U - the decay constants of ^{230}Pa and ^{230}U, and $BR_{Pa,U}$ - the branching ratio for the decay of ^{230}Pa to ^{230}U. In the script, these quantities are calculated by the user function *Production*():

```
for (t = 0; t <= time; t += time/10)          // loop over 10 time steps from 0 h to 50 h
{  APa = R_Pa230 * (1 - exp(-kPa * t));
   AU = R_Pa230*kU*kPa*BR_PaU * ((1-exp(-kPa*t))/kPa-(1-exp(-kU*t))/kU) / (kU – kPa);
   print("" + (t/3600) + ", " + (APa) + ", " + (AU));  }
```

In addition, the function outputs the results in a form suitable for the Nucleonica graph module (see Fig.3). The calculations show that, by the end of the 50 h irradiation, 80.5 GBq of ^{230}Pa together with 0.23 GBq of ^{230}U are produced.

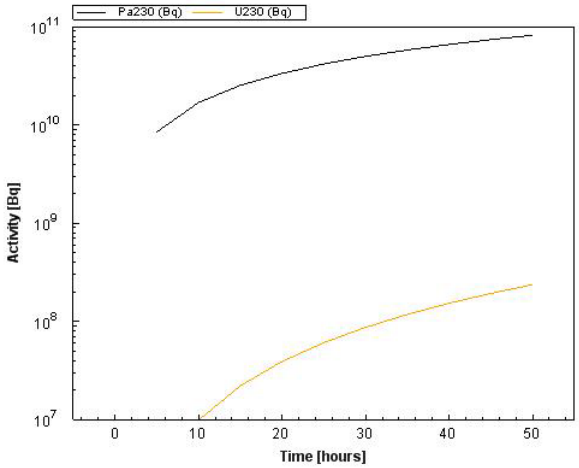

Fig 3. The production dynamics of ^{230}Pa and ^{230}U during 50 h proton activation of the Th foil

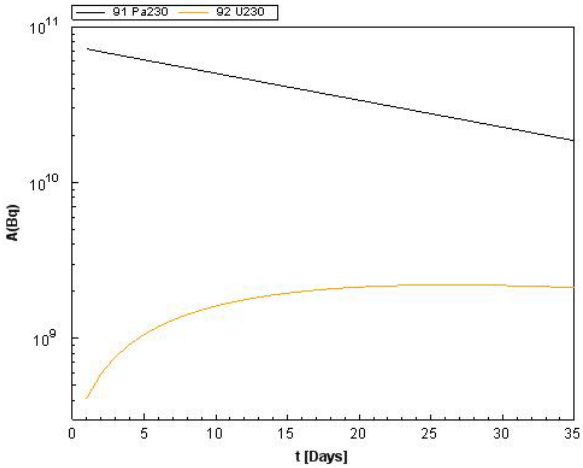

Fig 4. The 35-day-dynamics of ^{230}U accumulation in the decay of ^{230}Pa

3.2 Modelling the accumulation of ^{230}U

To model the accumulation of ^{230}U following the irradiation period, an object of the nuclide mixture class *vnuclide*, consisting of 80.5 GBq of ^{230}Pa and 0.23 GBq of ^{230}U, was created and the *vnuclide.Decay*() method was employed. Fig.4 demonstrates the obtained dynamics of the ^{230}U activity within the 35-day-long post-irradiation interval. One can see from the graph that the ^{230}U activity reaches its maximum in 24-30 days after the irradiation. The more accurate value t_{max} = 26.9 days for the optimum accumulation time interval has been obtained in the script by implementing the formula:

$$t_{\max} = \frac{1}{k_U - k_{Pa}} \ln\left(\frac{k_U}{k_{Pa}} + \frac{A_U(0)}{A_{Pa}(0)}\right).$$

From Fig.4, however, it follows that the actual separation of ^{230}U can be performed up to 2 weeks earlier without a significant loss in the resulting activity of the target nuclide (compare A_U(12 day) = 1.88 GBq and $A_U(t_{max})$ = 2.34 GBq). This process of in-growth and separation can be repeated as long as significant activity of ^{230}U can be produced.

4. Conclusions

The Nucleonica scripting is an object oriented language which is based on the physically intuitive classes, methods and properties. Through the script interface the user has access to the main Nucleonica applications and nuclear data. By combining results from multiple Nucleonica applications and nuclear data, users can go beyond the conventional single-application approach for solving their specific experimental tasks and problems.

5. References

[1] J. Magill et al., Nucleonica: A Nuclear Science Portal, ENS News, Issue No.17 Summer (July 2007), see www.euronuclear.org/e-news/e-news-17/nucleonica.htm
[2] J. Magill, Nucleonica: A Web Portal for the Nuclear Sciences, this conference.
[3] NucleonicaWiki: see www.nucleonica.net:81/wiki/index.php/Special:Allpages/Help:
[4] A. Morgenstern et al., Appl. Rad. Isotop. 2008, in press.